

Adding OpenMP

Heidi Poxon

Sr. Principal Engineer

Cray Programming Environment



CRAY®

Legal Disclaimer



Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publicly announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA and YARCDATA. The following are trademarks of Cray Inc.: CHAPEL, CLUSTER CONNECT, CLUSTERSTOR, CRAYDOC, CRAYPAT, CRAYPORT, DATAWARP, ECOPHLEX, LIBSCI, NODEKARE, REVEAL. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used on this website are the property of their respective owners.

Reveal



The screenshot displays the Cray Reveal application interface. On the left, a 'Navigation' pane shows a tree of loops, with 'Loop@51' selected. The main window shows the source code for 'vhone.pl' with a loop starting at line 51. A 'Scoping Results' window is open, showing a table of variables and their scopes. Below it, an 'OpenMP Directive' window shows the generated directives for the selected loop.

Name	Type	Scope	Info
f	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit. WARN: LastPrivate of array may be very expensive.
flat	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit.
p	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit. WARN: LastPrivate of array may be very expensive.
q	Array	Unresolved	FAIL: Last defining iteration not known for variable that is live on exit. WARN: LastPrivate of array may be very expensive.
delp1	Scalar	Private	
delp2	Scalar	Private	
deltx	Scalar	Private	
dtheta	Scalar	Private	
dvol	Array	Private	FAIL: incompatible with 'natural' scope. WARN: LastPrivate of array may be very expensive.
dx	Array	Private	FAIL: incompatible with 'natural' scope. WARN: LastPrivate of array may be very expensive.
dx0	Array	Private	FAIL: incompatible with 'natural' scope. WARN: LastPrivate of array may be very expensive.
e	Array	Private	FAIL: incompatible with 'natural' scope. WARN: LastPrivate of array may be very expensive.

```
! Directive inserted by Cray Reveal. May be incomplete.
!$OMP parallel do default(none)
!$OMP & unresolved (dvol,dx,dx0,e,flat,p,para,q,r,radius,stheta,svel, &
!$OMP & theta,u,v,w,xa,xa0)
!$OMP & private (l,j,k,m,n,delp1,delp2,delp3,shock,temp2,old_flat,onemif,hd, &
!$OMP & sinxD,gamfac1,gamfac2,dtheta,deltx,fracn,ekin)
!$OMP & shared (gamma,isz,js,ks,mypex,mypa2,ngeomz,nleitz,npez,nrightz, &
!$OMP & recv3,send4,dzd,zc,zc,zza)
```

- Reduce effort associated with adding OpenMP to MPI programs
- Get insight into optimizations performed by the Cray compiler
- Add OpenMP as a first step to parallelize loops that will target GPUs
- Track requests to memory and evaluate the bandwidth contribution of objects within a program for loop tuning

Approach to Adding Parallelism

1. Identify key high-level loops
2. Perform parallel analysis and scoping
3. Add OpenMP directive layer of parallelism
4. Analyze performance for further optimization, specifically vectorization of innermost loops
5. Port parallel loops to GPU with OpenMP target directives

The Problem – How Do I Parallelize This Loop?

- How do I know this is a good loop to parallelize?
- What prevents me from parallelizing this loop?
- Can I get help building a directive?

```
subroutine sweepz
...
do j = 1, js
  do i = 1, isz
    radius = zxc(i+mypez*isz)
    theta = zyc(j+mypey*js)
    do m = 1, npez
      do k = 1, ks
        n = k + ks*(m-1) + 6
        r(n) = recv3(1,j,k,i,m)
        p(n) = recv3(2,j,k,i,m)
        u(n) = recv3(5,j,k,i,m)
        v(n) = recv3(3,j,k,i,m)
        w(n) = recv3(4,j,k,i,m)
        f(n) = recv3(6,j,k,i,m)
      enddo
    enddo
    ...
    call ppmlr
    do k = 1, kmax
      n = k + 6
      xa(n) = zza(k)
      dx(n) = zdz(k)
      xa0(n) = zza(k)
      dx0(n) = zdz(k)
      e(n) = p(n)/(r(n)*gamm)+0.5 &
        *(u(n)**2+v(n)**2+w(n)**2)
    enddo
    call ppmlr
  ...
enddo
enddo
```

```
subroutine ppmlr

call boundary
call flatten
call paraset(nmin-4, nmax+5, para, dx, xa)

call parabola(nmin-4, nmax+4, para, p, dp, p6, p1, flat)
call parabola(nmin-4, nmax+4, para, r, dr, r6, r1, flat)
call parabola(nmin-4, nmax+4, para, u, du, u6, ul, flat)

call states(pl, ul, r1, p6, u6, r6, dp, du, dr, plft, ulft, &
  rlft, prgh, urgh, rrgh)
call riemann(nmin-3, nmax+4, gam, prgh, urgh, rrgh, &
  plft, ulft, rlft, pmid, umid)
call evolve(umid, pmid) ← contains more calls

call remap ← contains more calls

call volume(nmin, nmax, ngeom, radius, xa, dx, dvol)

call remap ← contains more calls

return
end
```

Loop Work Estimates

Gather loop statistics using the [Cray performance tools](#) and the [Cray Compiling Environment \(CCE\)](#) to determine which loops have the most work

- Helps identify high-level serial loops to parallelize
 - Based on runtime analysis, approximates how much work exists within a loop

Collect Loop Work Estimates

- Set up loop work estimates experiment with Cray compiler and Cray performance tools
 - `user@login> module load PrgEnv-cray perftools-lite-loops`
- Build program with Cray program library
 - `-h pl=full_path/program.pl`
- Run program to get loop work estimates

Example Loop Statistics

Table 2: Loop Stats by Function

Loop Incl Time Total	Loop Hit	Loop Trips Avg	Loop Trips Min	Loop Trips Max	Function=/.LOOP[.] PE=HIDE
8.995914	100	25	0	25	sweepy_.LOOP.1.li.33
8.995604	2500	25	0	25	sweepy_.LOOP.2.li.34
8.894750	50	25	0	25	sweepz_.LOOP.05.li.49
8.894637	1250	25	0	25	sweepz_.LOOP.06.li.50
4.420629	50	25	0	25	sweepx2_.LOOP.1.li.29
4.420536	1250	25	0	25	sweepx2_.LOOP.2.li.30
4.387534	50	25	0	25	sweepx1_.LOOP.1.li.29
4.387457	1250	25	0	25	sweepx1_.LOOP.2.li.30
2.523214	187500	107	0	107	riemann_.LOOP.2.li.63

View Source and Optimization Information

The screenshot displays the Reveal IDE interface. The main window shows the source code for `parabola.f90`. The code is as follows:

```
66  
67 do n = nmin, nmax  
68   deltaa(n) = ar(n) - al(n)  
69   a6(n)      = 6. * (a(n) - .5 * (al(n) + ar(n)))  
70   scrch1(n) = (ar(n) - a(n)) * (a(n) - al(n))  
71   scrch2(n) = deltaa(n) * deltaa(n)  
72   scrch3(n) = deltaa(n) * a6(n)  
73 enddo  
74  
75 do n = nmin, nmax  
76   if(scrch1(n) <= 0.0) then  
77     ar(n) = a(n)  
78     al(n) = a(n)  
79   endif
```

Optimization information is overlaid on the code. A green vertical bar on line 67 indicates a loop fusion with line 53. A blue vertical bar on line 75 is labeled `Vr2`. A bracket labeled `f` spans from line 67 to line 73. An `Info` window at the bottom states: "A loop starting at line 67 was fused with the loop starting at line 53."

The left sidebar shows a **Navigation** pane with a **Loop Performance** table:

Time	Label	Stars
4.0423	SWEEPX2@32	★
3.8576	SWEEPZ@51	★
3.8573	SWEEPZ@52	★
2.2068	RIEMANN@63	★
1.2299	RIEMANN@64	★
0.8068	PARABOLA@67	
0.0146	Instance #1	
0.0156	Instance #2	
0.0156	Instance #3	
0.0163	Instance #4	
0.0163	Instance #5	
0.0174	Instance #6	
0.0167	Instance #7	

Below the table is a **Traceback** pane showing the call stack:

- PARABOLA@67
- PPMLR@51
- sweepx1_.LOOP.2.li.32@53
- sweepx1_.LOOP.1.li.31@32
- SWEEPX1@31
- VHONE@232

The **Loopmark Legend** window on the right lists various optimization markers:

- A** Pattern Matched
- C** Collapsed: A loop nest has been collapsed into one loop
- D** Deleted
- E** Cloned
- G** Accelerated
- I** Inlined
- II** Not Inlined
- L** Loop
- M** Multithreaded: A loop or block of code is multi-threaded
- R** Region
- S** Scoping Analysis
- V** Vectorized
- a** Atomic Memory Operation
- b** Blocked
- c** Conditional and/or Computed
- f** Fused
- g** Partitioned
- i** Interchanged
- n** Non-blocking Remote Transfer
- p** Partial
- r** Unrolled
- s** Shortloop: A loop was converted to a single vector iteration
- w** Unwound

Scope Selected Loop(s)

The screenshot shows a window titled "Reveal OpenMP Scoping" with two tabs: "Scope Loops" and "Scoping Results". The "Scoping Results" tab is active, displaying a table of loops to be scoped. The table has three columns: "Scope?", "Line #", and "File or Source Line". The "Scope?" column contains checkboxes, and the "Line #" column is highlighted in green. The "File or Source Line" column lists various source files and line numbers. At the bottom of the window, there are controls for "Apply Filter", "Time", "Trips", "Threads", and "Speedup". A "Start Scoping" button is highlighted, and a status bar at the bottom indicates "26 Loops selected".

Scope?	Line #	File or Source Line
<input checked="" type="checkbox"/>		/home/users/heidi/reveal/evolve.f90
<input type="checkbox"/>		/home/users/heidi/reveal/flatten.f90
<input type="checkbox"/>		/home/users/heidi/reveal/forces.f90
<input type="checkbox"/>		/home/users/heidi/reveal/images.f90
<input type="checkbox"/>		/home/users/heidi/reveal/init.f90
<input type="checkbox"/>		/home/users/heidi/reveal/parabola.f90
<input type="checkbox"/>		/home/users/heidi/reveal/ppmlr.f90
<input type="checkbox"/>		/home/users/heidi/reveal/prin.f90
<input checked="" type="checkbox"/>		/home/users/heidi/reveal/remap.f90
<input checked="" type="checkbox"/>		/home/users/heidi/reveal/riemann.f90
<input checked="" type="checkbox"/>		/home/users/heidi/reveal/states.f90
<input checked="" type="checkbox"/>		/home/users/heidi/reveal/sweepx1.f90
<input checked="" type="checkbox"/>		/home/users/heidi/reveal/sweepx2.f90
<input checked="" type="checkbox"/>		/home/users/heidi/reveal/sweezy.f90
<input checked="" type="checkbox"/>		/home/users/heidi/reveal/sweepz.f90

Apply Filter Time: 0.000 Trips: 2 Threads: 4 Speedup: 0.010

Start Scoping Cancel 26 Loops selected Close

- Trigger dependence analysis
- scope loops above given threshold

Review Scoping Results

Loops with scoping information are flagged. Red needs user assistance

```
4.0778 SWEEPY@35 ★
4.0773 SWEEPY@36 ★
4.0529 SWEEPX1@31 ★
4.0526 SWEEPX1@32 ★
4.0425 SWEEPX2@31 ★
4.0423 SWEEPX2@32 ★
3.8576 SWEEPZ@51 ★
3.8573 SWEEPZ@52 ★
2.2068 RIEMANN@63 ★
1.2299 RIEMANN@64
0.8068 PARABOLA@67
0.5429 PARABOLA@44
0.5331 PARABOLA@75
0.4244 REMAP@83 ★
0.3341 PARABOLA@30
0.2966 PARABOLA@84
0.2915 PARABOLA@53
0.2287 RIEMANN@44 ★
0.2028 PARABOLA@36
0.2009 PARABOLA@117
0.1858 PARABOLA@24
0.1847 SWEEPY@86 ★
0.1771 STATES@64 ★
0.1723 EVOLVE@70 ★
0.1638 REMAP@111 ★
0.1619 PARABOLA@129
0.1070 PARABOLA@139
0.0938 SWEEPZ@120 ★
0.0936 SWEEPZ@121 ★
0.0930 SWEEPZ@122 ★
0.0925 SWEEPX1@59 ★
0.0901 SWEEPZ@22 ★
0.0898 SWEEPZ@23 ★
0.0892 STATES@50
0.0880 SWFFP7@105 ★
```

```
52 do i = 1, isz
53   radius = zxc(i+mypez*isz)
54   theta = zyc(j+mypez*js)
55   stheta = sin(theta)
56   radius = radius * stheta
57
58 ! Put state variables into 1D arrays, padding with 6 ghost zones
59 do m = 1, npez
60   do k = 1, ks
61     n = k + ks*(m-1) + 6
62     r(n) = recv3(1,j,k,i,m)
63     p(n) = recv3(2,j,k,i,m)
64     u(n) = recv3(5,j,k,i,m)
65     v(n) = recv3(3,j,k,i,m)
66     w(n) = recv3(4,i,k,i,m)
```

Info - Line 51

- A loop starting at line 51 was scoped with errors. See Scoping Tool for more information. "ppmlr" (called from "sweepz") was not inlined because I/O was detected in "volume".
- "ppmlr" (called from "sweepz") was not inlined because the enclosing loop body did not completely flatten.
- A loop starting at line 105 is flat (contains no external calls).
- A loop starting at line 105 was not vectorized because it does not map well onto the target architecture.
- A loop starting at line 105 was unrolled 8 times.
- A loop starting at line 51 was not vectorized because it contains a call to subroutine "ppmlr" on line 81.
- A loop starting at line 52 was not vectorized because it contains a call to subroutine "ppmlr" on line 81.
- A loop starting at line 59 is flat (contains no external calls).
- A loop starting at line 59 was not vectorized because a better candidate was found at line 60.
- A loop starting at line 60 is flat (contains no external calls).
- A loop starting at line 60 was not vectorized because it does not map well onto the target architecture.
- A loop starting at line 60 was unrolled 8 times.
- A loop starting at line 71 is flat (contains no external calls).
- A loop starting at line 71 was vectorized.

sweepz.f90: Loop@51

Name	Type	Scope	Info
wl@remap_I	Scalar	Unresolved	FAIL: Possible recurrence involving this object.
xa	Array	Unresolved	FAIL: Possible resolvable recurrence involving this object. FAIL: Possible resolvable recurrence involving this object. WARN: LastPrivate of array may be very expensive.
xa0	Array	Unresolved	FAIL: Possible recurrence involving this object. FAIL: Possible resolvable recurrence involving this object. WARN: LastPrivate of array may be very expensive.
i	Scalar	Private	
j	Scalar	Private	
k	Scalar	Private	
m	Scalar	Private	
n	Scalar	Private	
stheta	Scalar	Private	
theta	Scalar	Private	
gamm	Scalar	Shared	
isz	Scalar	Shared	
js	Scalar	Shared	
ks	Scalar	Shared	
mypey	Scalar	Shared	

Parallelization inhibitor messages are provided to assist user with analysis

Review Scoping Results (continued)

The screenshot shows the 'Reveal OpenMP Scoping' application window. The title bar reads 'Reveal OpenMP Scoping'. The interface has two tabs: 'Scope Loops' and 'Scoping Results', with 'Scoping Results' selected. The main area displays 'sweepy.f90: Loop@35'. A call chain is shown in a yellow box: 'Call or I/O at line 62 of sweepy.f90', '4: /home/users/heidi/reveal/volume.f90:34', and '3: /home/users/heidi/reveal/evolve.f90:21'. Below this is a table with columns 'Name', 'Type', 'Scope', and 'Info'. The table lists variables like 'ks', 'mypey', 'ndim', 'npey', 'recv1', 'send2', 'svel', 'zdy', 'zxc', and 'zya'. The 'svel' row is highlighted in red and has a 'WARN' message: 'WARN: atomic reduction operator required unless reduction fully'. Two callouts provide context: one points to the call chain with the text 'Reveal identifies calls that prevent parallelization', and another points to the 'svel' row with the text 'Reveal identifies shared reductions down the call chain'. At the bottom, there are buttons for 'Insert Directive', 'Show Directive', and 'Close', along with a 'Find Name:' search field.

Name	Type	Scope	Info
ks	Scalar	Shared	
mypey	Scalar	Shared	
ndim	Scalar	Shared	
npey	Scalar	Shared	
recv1	Array	Shared	
send2	Array	Shared	
svel RI	Scalar	Shared	WARN: atomic reduction operator required unless reduction fully
zdy	Array	Shared	
zxc	Array	Shared	
zya	Array	Shared	

Review Scoping Results (continued)

Name	Type	Scope	Info
a0i	Scalar	Private	
a0r	Scalar	Private	
a1i	Scalar	Private	
a1r	Scalar	Private	
a2i	Scalar	Private	
a2r	Scalar	Private	
b0i	Scalar	Private	
b0r	Scalar	Private	
b1i	Scalar	Private	
b1r	Scalar	Private	
b2i	Scalar	Private	
b2r	Scalar	Private	
j	Scalar	Private	
a	Scalar	Shared	WARN: Assuming no overlap with other objects. INFO: additional detail.
b	Scalar	Shared	WARN: Assuming no overlap with other objects. INFO: additional detail.
c	Scalar	Shared	WARN: Assuming no overlap with other objects. INFO: additional detail.

Assume no overlap between lattice[*].mom[*] and tempmom[*][*]

Review Scoping Results (continued)

Reveal OpenMP Scoping

Scope Loops | Scoping Results

fluxk.f: Loop@28

Name	Type	Scope	Info
fsk	Array	Private	WARN: LastPrivate of array may be very expensive. FAIL: FirstPrivate/Shared Scope Conflict.
fsk I	Array	Private	FAIL: FirstPrivate/Shared Scope Conflict.
i	Scalar	Private	
j	Scalar	Private	
k	Scalar	Private	
l	Scalar	Private	FAIL: Ambiguous store conflict.
qs	Array	Private	WARN: LastPrivate of array may be very expensive. FAIL: Last defining iteration not known for variable that may be live on exit.
qsp	Scalar	Private	
qspk	Scalar	Private	
dq	Array	Shared	
dtv	Array	Shared	
ind	Scalar	Shared	FAIL: conflicting requirements, unable to scope.
jcmax	Scalar	Shared	
kadd	Scalar	Shared	

First/Last Private: Enable FirstPrivate Enable LastPrivate

Reduction:

Find Name:

View Loops through Call Chain

Navigation: Loop Performance

- 4.0778 SWEEPY@35
- 4.0773 SWEEPY@36
- 4.0529 SWEEPX1@31
- 4.0526 SWEEPX1@32
- 4.0425 SWEEPX2@31
- 4.0423 SWEEPX2@32
- 3.8576 SWEEPZ@51
- 3.8573 SWEEPZ@52
- 2.2068 RIEMANN@63
- 0.3584 PPMLR@73
- 0.3566 PPMLR@73
- 0.3566 PPMLR@73
- 0.3866 PPMLR@73
- 0.3909 PPMLR@73
- 0.3576 PPMLR@73
- 1.2299 RIEMANN@64

Source - /ufs/home/users/heidi/reveal/riemann.f90

```
62
63 do l = lmin, lmax
64   do n = 1, 12
65     pmold(l) = pmid(l)
66     wlft(l) = 1.0 + gamfac1*(pmid(l) - plft(l)) * plfti(l)
67     wrgh(l) = 1.0 + gamfac1*(pmid(l) - prgh(l)) * prghi(l)
68     wlft(l) = clft(l) * sqrt(wlft(l))
69     wrgh(l) = crgh(l) * sqrt(wrgh(l))
70     zlft(l) = 4.0 * vlft(l) * wlft(l) * wlft(l)
71     zrgh(l) = 4.0 * vrgh(l) * wrgh(l) * wrgh(l)
72     zlft(l) = -zlft(l) * wlft(l)/(zlft(l) - gamfac2*(pmid(l) - plft(l)))
73     zrgh(l) = zrgh(l) * wrgh(l)/(zrgh(l) - gamfac2*(pmid(l) - prgh(l)))
74     umidl(l) = ulft(l) - (pmid(l) - plft(l)) / wlft(l)
75     umidr(l) = urgh(l) + (pmid(l) - prgh(l)) / wrgh(l)
76     pmid(l) = pmid(l) + (umidr(l) - umidl(l))*(zlft(l) * zrgh(l)) /
77               (zlft(l) - gamfac2*(pmid(l) - plft(l)) + zrgh(l) - gamfac2*(pmid(l) - prgh(l)))
78   end do
79 end do
80 if (abs(pmid(l)-pmold(l))/pmid(l) < tol) exit
```

Info - Line 63
● A loop starting at line 63 was not vectorized for an unspecified reason.

Traceback
PPMLR@73
sweepy_LOOP.2.li.36@67
sweepy_LOOP.1.li.35@36
SWEEPY@35
sweepy_LOOP.1.li.35@36
SWEEPY@35
VHONE@237

vhone.pl loaded. vhone_loops.ap2 loaded.

Generate OpenMP Directives

```
! Directive inserted by Cray Reveal. May be incomplete.
!$OMP parallel do default(none) &
!$OMP& unresolved (dvol,dx,dx0,e,f,flat,p,para,q,r,radius,svel,u,v,w, &
!$OMP& xa,xa0) &
!$OMP& private (i,j,k,m,n,$$ _n,delp2,delp1,shock,temp2,old_flat, &
!$OMP& onemfl,hdt,sinxf0,gamfac1,gamfac2,dtheta,deltx,fractn, &
!$OMP& ekin) &
!$OMP& shared (gamm,isy,js,ks,mypey,ndim,ngeomy,nlefty,npey,nrighty, &
!$OMP& recv1,send2,zdy,zxc,zya)
do k = 1, ks
do i = 1, isy
radius = zxc(i+mypey*isy)

! Put state variables into 1D arrays, padding with 6 ghost zones
do m = 1, npey
do j = 1, js
n = j + js*(m-1) + 6
r(n) = recv1(1,k,j,i,m)
p(n) = recv1(2,k,j,i,m)
u(n) = recv1(4,k,j,i,m)
v(n) = recv1(5,k,j,i,m)
w(n) = recv1(3,k,j,i,m)
f(n) = recv1(6,k,j,i,m)
enddo
enddo

do j = 1, jmax
n = j + 6
```

Reveal generates OpenMP directive with illegal clause marking variables that need addressing

Validate User Inserted Directives

Navigation

- ▶ images.f90
- ▶ init.f90
- ▶ parabola.f90
- ▶ ppm1r.f90
- ▶ prin.f90
- ▶ remap.f90
- ▼ riemann.f90 ■
 - ▼ RIEMANN ■
 - Loop@44
 - Loop@69 ■
 - Loop@70 ▲
 - Loop@89
- ▶ states.f90
- ▶ sweepx1.f90
- ▶ sweepx2.f90
- ▶ sweepy.f90
- ▶ sweepz.f90
- ▶ vh1.mods.f90
- ▶ vhone.f90
- ▶ volume.f90
- ▶ zonemod.f90

Source - /ufs/home/users/heidi/reveal/riemann.f90

```
63 : Directive inserted by Cray Reveal. May be incomplete.  
64 !$OMP parallel do default(none)  
65 !$OMP& private (l)  
66 !$OMP& shared (lmin,lmax,prgh,urgh,vrgh,plft,ulft,vlft,pmid,clft,  
67 !$OMP& crgh,gamfac1,gamfac2,plfti,pmold,prghi,umidl,umidr  
68 !$OMP& wlft,wrgh,zlft,zrgh,n)  
69 do l = lmin, lmax  
70 do n = 1, 12  
71 pmold(l  
72 wlft (l  
73 wrgh (l  
74 wlft (l  
75 wrgh (l  
76 zlft (l
```

Info - Line 69

- A loop starting at line 69 was not parallelized.
- A loop starting at line 69 was parallelized.

Scope Loops

Name	Type	Scope	Info
l	Scalar	Private	
n	Scalar	Private	WARN: Scope does not agree with user OMP directive.
clft	Array	Shared	
crgh	Array	Shared	
gamfac1	Scalar	Shared	
gamfac2	Scalar	Shared	

First/Last Private:
 Enable FirstPrivate
 Enable LastPrivate

Reduction:
None

Find Name:

vhone.pl loaded

User inserted directive with mis-scoped variable 'n'

Look For Vectorization Opportunities

Choose "Compiler Messages" view to access message filtering, then select desired type of message

The screenshot shows a compiler interface with a 'Navigation' pane on the left and a 'Source' pane on the right. The 'Navigation' pane is set to 'Compiler Messages' and shows a tree view of files and lines. The 'Source' pane displays Fortran code with line numbers 62-79. Line 64 is highlighted in green, and line 77 is highlighted in red. An 'Info' pane at the bottom provides details about the highlighted lines.

```
62  
63 do l = lmin, lmax  
64 do n = 1, 12  
65   pmold(l) = pmid(l)  
66   wlft(l) = 1.0 + gamfac1*(pmid(l) - plft(l)) * plfti(l)  
67   wrgh(l) = 1.0 + gamfac1*(pmid(l) - prgh(l)) * prghi(l)  
68   wlft(l) = clft(l) * sqrt(wlft(l))  
69   wrgh(l) = crgh(l) * sqrt(wrgh(l))  
70   zlft(l) = 4.0 * vlft(l) * wlft(l) * wlft(l)  
71   zrgh(l) = 4.0 * vrgh(l) * wrgh(l) * wrgh(l)  
72   zlft(l) = -zlft(l) * wlft(l)/(zlft(l) - gamfac2*(pmid(l) - plft(l))  
73   zrgh(l) = zrgh(l) * wrgh(l)/(zrgh(l) - gamfac2*(pmid(l) - prgh(l))  
74   umidl(l) = ulft(l) - (pmid(l) - plft(l)) / wlft(l)  
75   umidr(l) = urgh(l) + (pmid(l) - prgh(l)) / wrgh(l)  
76   pmid(l) = pmid(l) + (umidr(l) - umidl(l))*zlft(l) * zrgh(l) / (  
77   pmid(l) = max(smallp,pmid(l))  
78   if (abs(pmid(l)-pmold(l))/pmid(l) < tol ) exit  
79 enddo
```

Info - Line 64

- A loop starting at line 64 is flat (contains no external calls).
- A loop starting at line 64 was not vectorized because a recurrence was found on "pmid" at line 77.

QUESTIONS?

